Muller C-element based Decoder (MCD): A Decoder Against Transient Faults

Yangyang Tang*†, Emmanuel Boutillon*, Chris Winstead†, Christophe Jégo‡ and Michel Jézéquel§
*Université de Bretagne Sud, UMR CNRS 6285 Lab-STICC, Lorient, France
†Dept. of Electrical and Computer Engineering, Utah State University, Logan, Utah 84322
‡Institut Polytechnique Bordeaux, UMR CNRS 5218 Lab-IMS, Bordeaux, France
§Institut TELECOM/TELECOM Bretagne, UMR CNRS 6285 Lab-STICC, Brest, France
Email: yangyang.tang@univ-ubs.fr

***Abstract*— This work extends the analysis and application of a digital error correction method called Muller C-element Decoding (MCD), which has been proposed for fault masking in logic circuits comprised of unreliable elements. The proposed technique employs cascaded Muller C-elements and XOR gates to achieve efficient error-correction in the presence of internal upsets. The error-correction analysis of MCD architecture and the investigation of C-element's robustness are first introduced. We demonstrate that the MCD is able to produce error-correction benefit in a high error-rate of internal faults. Significantly, for a (3,6) short-length Low Density Parity Check (LDPC) code, when the decoding process is internally error-free the MCD achieves also a gain in terms of decoding performance by comparison to the well-known Gallager Bit-Flipping method. We further consider application of MCD to a general-purpose fault-tolerant model, coded Dual Modular Redundancy (cDMR), which offers low-redundancy error-resilience for contemporary logic systems as well as future nanoeletronic architectures.**

I. Introduction

Due to the rapid development of logic circuit manufacturing in the last two decades, electronic devices are now miniaturized to nanoscale dimensions. Nanoscale devices are increasingly sensitive to faults due to manufacturing defects, environmental fluctuations, and electronic noise or interference. From the circuit perspective, semiconductor faults fall into three main categories: permanent, intermittent, and transient. Permanent faults are caused by manufacturing defects, or device wear-out. They reflect irreversible physical changes. Intermittent faults occur because of unstable or marginal hardware; they can be activated and later reversed by environmental changes, like higher or lower temperature and voltage. Transient defects occur on a shorter time-scale, arising from device noise, interference or particle interactions. Due to the size downscaling processes, digital logic circuits are increasingly vulnerable to the transient faults [1].

Since any embedded fault-masking solution is implemented using the same device as the logic it protects, it is important for a fault-masking technique to be also tolerant of internal errors within its own logic. Researchers previously investigated the performance of a variety of error-correcting schemes under the influence of intrinsic faults [2]–[7]. In many cases, the proposed fault-masking solutions are limited to particular applications or algorithms. For example, an Algorithmic Noise-Tolerance (ANT) technique was introduced by Shim *et al.*, for reliable low-power digital signal processing [2]. The ANT method is based on a reduced precision replica to achieve low-power robust system in certain DSP algorithms. However, the ANT method is difficult to adapt if the reduced precision block cannot be done in an efficient way. An Error Resilient System Architecture (ERSA) [3] was proposed for low-cost robust systems, but incurs higher costs for general-purpose applications. Thus, the need of an efficient general-purpose fault-tolerant design is evident.

In the authors' previous work [8], [9], an error-correction method based on Muller C-elements was introduced and shown to be robust against internal transient faults. This method, referred as to MCD, is an implementation of iterative Low Density Parity Check (LDPC) [10] decoders suitable for embedding into digital logic circuits. The MCD method consists primarily of C-elements and XOR gates which implement a parity-check decoding algorithm. The MCD differs from traditional parity-check algorithms in that its circuit topology is designed to suppress internal upsets within the decoder.

In this paper, an extended work of MCD technique and its implementation for embedded robust design are detailed. The main contributions of this work are threefold: First the error-correction analysis of MCD is shown. Second, the study of error-resilience capacity inherited by C-element is detailed. Lastly, we study a coded Dual Modular Redundancy (cDMR) application with a short length MCD, which achieves a significant gain in terms of decoding performance for both error-free and noisy decoding cases.

II. Bit-Flipping Method and MCD Architecture

A. Gallager's Bit-Flipping Method

LDPC decoding methods are traditionally described as message-passing on a code's Tanner graph [11]. The Tanner graph contains two sets of nodes – variable nodes v_i and parity-check nodes p_j. Some edges connect the variable nodes to the parity-check nodes thanks to a sparse parity-check matrix associated to each LDPC code. The degree of a node is the number of edges connected to it in the Tanner graph. Let d_v be the degree of variable node v_i, and let $\mathcal{V}_i$ be the set of edges that are connected to v_i. Similarly, let d_c be the degree of parity-check node p_j, and let $\mathcal{P}_j$ be the set of edges that are connected to p_j. During the decoding process, binary

978-1-4673-5762-3/13/$31.00 ©2013 IEEE

messages are exchanged between the two sets of nodes. The message passed from node v_i to p_j is written y_{ij}, and the returning message from p_j to v_i is written f_{ji}. All nodes are simple processing functions that follow the standard extrinsic information principle: the outgoing message on edge k is computed using information from all edges except k.

Gallager's Bit-Flipping Methods (GBF) were among the first LDPC decoding algorithms introduced by Gallager [10]. The GBF methods are defined for frames that were transmitted over a Binary Symmetric Channel (BSC), as a binary hard decision message-passing decoding method [12]. A GBF algorithm can be described as follows. The variable nodes initially transmit messages $y_{ij} = x_i$ for all i, j. For each parity-check node, the outgoing message on edge k is equal to the modulo-2 sum of all local incoming messages (excluding edge k). For the variable nodes, the outgoing message along edge k is equal to x_i unless at least b incoming messages (excluding edge k) disagree with the x_i. Traditionally, there are two versions of the algorithm. When $b = (d_v - 1)$, all local incoming messages are required to be unanimous, the method is known as the *Gallager-A* algorithm. If b is fixed to a smallest integer during the decoding iteration as explained in [10] and [12], then the method is known as the *Gallager-B* algorithm.

B. The MCD Architecture

The MCD architecture is characterized by the processing steps applied at the variable nodes. For each variable node v_i, a set of C-element gates C_k, $0 \leq k < (d_v - 1)$, is associated as shown in Fig. 1(a) [8]. The circuit is a cascade of C-element gates, modified for the initialization of the state memory. In this figure, $d_v = 4$. Each C-element has three inputs; the left-side inputs are the usual inputs, and the top-side input is the initial state for the C-element's memory. Each C-element gate C_k contains a single-bit storage element c_k. Moreover, a standard binary C-element circuit is shown in Fig. 1(b) [13]. The MCD decoding algorithm is described as follows:

1) Initialize $y_k = x_i$, for all $k \in \mathcal{V}$.
2) Compute $f_{ji} = \oplus_{m \in P_{j \setminus i}} y_{mj}$ for all $j \in \mathcal{P}$.
3) Initialize each C-element memory as $c_k = f_{k'}$, where $k' = (k + d_v - 1) \bmod d_v$.
4) The C-element's port connections are as follows. For C_0, the inputs are f_0 ($f_0 = x_i$) and f_1, and the output is c_0. For C_k, the inputs are c_{k-1} and f_{k+1}, and the output is c_k.
5) Iterate steps 2 and 4 during a fixed number of iterations, as the restoration phase. Note that the initialization in step 3 is performed only during the first iteration.
6) The corrected output is $z_i = c_{(d_v - 1)}$.

This algorithm is guaranteed to correct a single error during each iteration. It can also correct many multi-error patterns as well [8], [9]. The MCD method can be regarded as a circuit-level implementation for variable nodes in an LDPC decoder.

III. Error-Correction Analysis

An error obtained from f_k may originate from an internal error within the decoder. The C-element cascade helps to stop the propagation of such errors. As the error-correction analysis done in [9], a single error event can be masked during an iteration. In the rest of this subsection we study the cases of double-error events in restoration phase during an iteration.

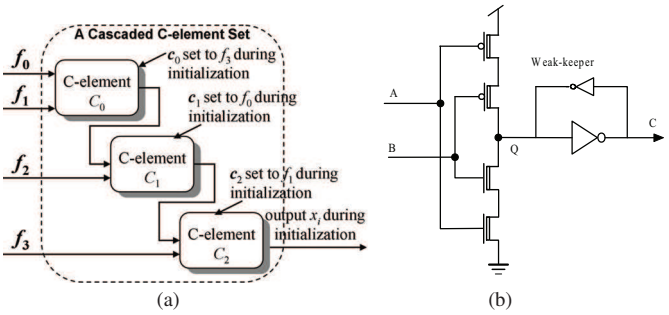


Fig. 1: A local implementation of a variable node (a), and a standard Muller C-element circuit of which the state memory c_k is realized by a weak-keeper (b).

A. Double errors appear at f_k and f_l

If a variable node v_i receives two erroneous messages at f_k and f_l, such that $|k - l| \geq 1$, there are three possible cases:

- If f_0 and f_1 are erroneous, the output of C_0 is thus erroneous. Since the rest of f_k and storages are correct, the error from C_0 is masked.
- f_0 and f_k ($k > 1$) are incorrect. In this case, the output of C_0 is correct, and any erroneous f_k can't induce an error event.
- f_k and f_l, ($k \neq l \neq 0$), are erroneous. In this case, any single error event is waived.

B. Double errors appear at c_k and c_l

If a variable node v_i generated two upsets at c_k and c_l, such that $|k - l| \geq 1$, these events can be sorted as a single error event occurrence. Consequently, these cases are masked by the inherent fault-tolerance by C-element.

C. Double errors appear at f_k and c_l

If two upsets occur at f_k and c_l of a variable node v_i, where $|k - l| \geq 0$, three cases are possible:

- if $k = (d_v - 1)$ and $l \neq (d_v - 2)$, the value of C-element C_l, c_l, is correct. Since only single-error event is occurred among the C-elements except C_l, due to C-element's behavior another input of C_l is assured error-free. In this case, only f_k is erroneous that can not induce an error at the output.
- When $k \neq (d_v - 1)$ and $l \neq (d_v - 2)$, f_k and c_l are correct. Hence, the output from v_i is correct as well.
- At last, if $k = (d_v - 1)$ and $l = (d_v - 2)$, f_k and c_l are thus incorrect. In this case, the output is erroneous.

To sum up, a cascaded C-element set is able to correct any single error event regardless of where those errors originate. Any double-error events can be corrected, except a single pattern, namely the local incoming message to the last C-element and its storage are simultaneously flipped over. For the burst errors, the C-element set can also correct some patterns.

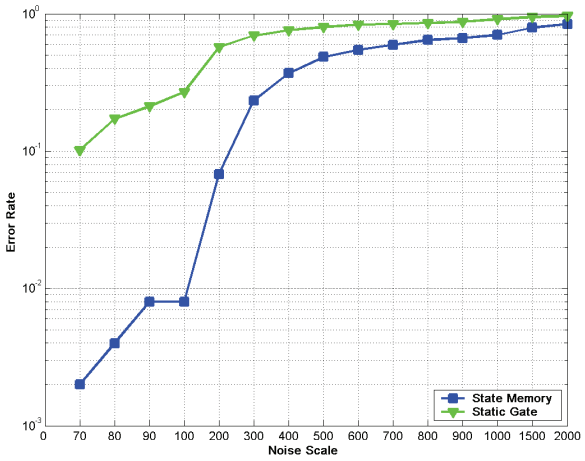


Fig. 2: Simulated error rate results obtained using the $0.6\mu m$ CMOS model in Virtuoso Spectre.

IV. Fault-Tolerance Inherited by C-element

To further reveal the error-resilience inherited from C-element, we study the reliability of a state memory of C-element by comparison to a static logic gate, namely an inverter. Without loss of generality, the state memory was designed for a $0.6\mu m$ CMOS logic process. For comparison, an inverter was also designed. Both circuits were simulated in Virtuoso Spectre from Cadence, where signal errors are set up in a large scale by the "noisescale" parameter. Fig. 2 shows an overlay of Monte Carlo transient simulation runs form both the state memory and the static gate simulations. The "noisescale" parameter is set as large scale in order to induce enough upset cases to measure and compare the error rate for those two components. An error occurs whenever the difference crosses $0.05V$ threshold when $5V$ as the correct output. More precisely, if the difference of output value to the correct one $(5V)$ is bigger than $0.05V$, then an error is considered. The difference lower than $0.05V$ is bypassed. For the static gate, errors appear quite frequently as indicated by numerous threshold-crossings. By contrast, in the state memory case, the amplitude of output noise fluctuations is significantly cutoff. Thanks to the feedback mechanism in the state memory, up to a magnitude of two orders as the robustness gain. Consequently, by comparison to a static logic gate, C-element is able to tolerates the upset events with error-resilience gain between multiple times up to magnitude of two orders.

V. The cDMR Technique by Applying MCD

A. cDMR Technique

One of the authors (Winstead) proposed an LDPC-coded Fault Compensation Technique (LFCT) in [5]. This technique is relevant to higher LDPC codes resulting in a more powerful error-correcting ability. With taking the principle of LFCT, a coded Dual-Modular Redundancy (cDMR) technique can be defined as shown in Fig. 3. More detailed can also be found

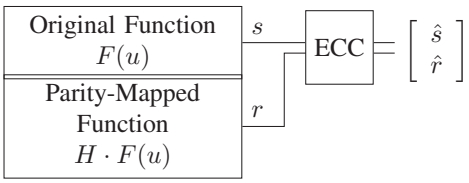


Fig. 3: Architecture of the cDMR model.

in [8] and [9]. A logic function $F(u)$ is implemented using a digital CMOS technology that is subject to errors at its output. The original function $F(u)$ is augmented by the addition of a redundant parity-generator module, $H \cdot F(u)$, where H represents the encoding function that generates parity bits codeword space at the output of $F(u)$ as explained in [5]. The *systematic* output word s is then concatenated with the parity outputs r from $H \cdot F(u)$, yielding a complete codeword $[s\ r]$. According to the code's H matrix, namely the parity-check matrix that defines the error-correction code, an Error Control Codes (ECC) is supposed to perform the error-correction in the presence of internal faults.

B. Good Decoder Candidate for the ECC of cDMR

In this subsection, we demonstrate a good candidate for the ECC of cDMR. By comparison to the GBF approach, the performance improvement achieved with MCD was analyzed using two metrics: the BER performances over BSC under a error-free decoding process and faulty decoding process. For the sake of facility, only LDPC code with $d_v = 3$ is studied.

A (3,6) LDPC code of length 64 was simulated over BSC in the cases of error-free and noisy decoding process, respectively, as shown in Fig. 4. First, in the case of noisy decoding as dashed curves, the GBF performance worsens with increased iterations, but MCD does not suffer from this degradation. Significantly, when the length of a LDPC code is short, the MCD performs a gain in terms of decoding performance, as shown the solid curves in Fig. 4. As such, this short MCD decoder is a good candidate to the ECC block. Consequently, with employing the MCD architecture, the cDMR technique that can prevent the internal upsets provides a general-purpose robust system to logic design.

VI. Discussion: cDMR's Implementation

To avoid the occurrence of correlated errors among s and r, the most reliable approach is regarded in two cases: if $F(u)$ has low complexity, such like finite-state machines, $F(u)$ and $H \cdot F(x)$ then can be easily designed without resulting correlated errors; if $F(u)$ has a high complexity and large fanin and fanout, a flat truth-table synthesis can be used, as is done with cross-bar logic arrays [14]. To explain this constraint, we may contrast the two circuits shown in Fig. 5. In Fig. 5(a), a ripple-carry implementation, a single gate error may propagate to the several of the output signals. An example of error propagation is indicated by the $\star$ symbol in Fig. 5(a). In the cDMR system, error-propagation may induce many simultaneous faults in the $[s\ r]$ codeword, which are not likely to be correctable.

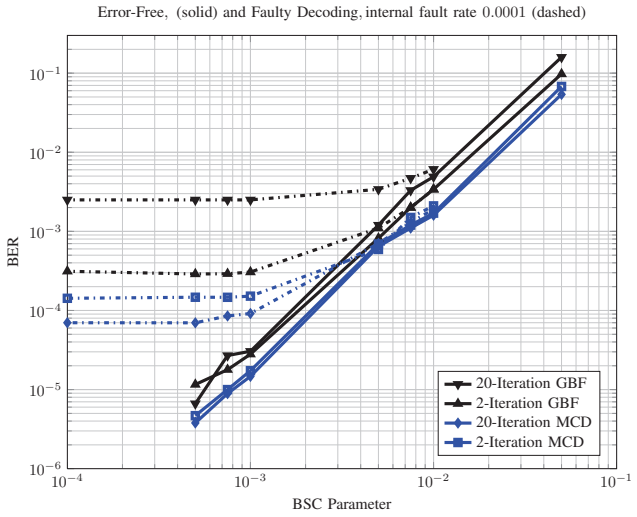
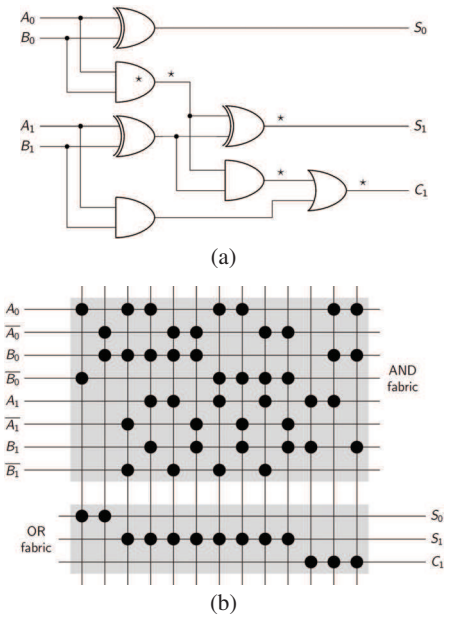


Fig. 4: Simulation results of (3,6) LDPC code of length 64 under faulty and error-free decoding.



(a)

(b)

Fig. 5: Two-bit binary adder implementations, representing a traditional ripple-carry design (a), and a crossbar design suitable for some nanoelectronic device families (b).

In the crossbar implementation, as shown in Fig. 5(b), logic is implemented by fabrics of AND-logic and OR-logic. The "dots" indicate the placement of junctions which physically implement the logic operations. In this style of implementation, every operation is associated with only a single output. If a momentary fault occurs at some junction, it will propagate only to a single output. This implementation guarantees that single-error events are correctable. The major disadvantage of crossbar logic is that the operation counts are not optimal. The crossbar adder in Fig. 5(b), for instance, is composed of 57 separate operations. Crossbar logic does not generally obtain minimized gate complexity, but it offers improved reliability by eliminating error propagations.

VII. Conclusion

The Muller C-element based Decoder, referred as to MCD, has been proposed for efficient error-correction in the presence of high error rate internal upsets. In this work, an extended study of MCD is presented. The error-correction analysis of MCD and the study of C-element's fault-tolerance are first introduced. By comparison to the well-known *Gallager's Bit-Flipping* method, for a (3,6) LDPC code of short length, the MCD yields a gain in terms of decoding performance in the cases of error-free and faulty decoding. At last, by applying the MCD, a coded Dual-Modular Redundancy (cDMR) technique designed for robust hard-on to current logic or future nanoelectronics is optimized.

Acknowledgment

This work was supported by the US National Science Foundation under award ECCS-0954747 and CCF-0916105.

References

[1] R. C. Baumann, "Soft errors in advanced semiconductor devices-part i: the three radiation sources," *IEEE Transactions on Device and Materials Reliability*, vol. 1, no. 1, pp. 17–22, 2001.

[2] B. Shim and N. Shanbhag, "Energy-efficient soft error-tolerant digital signal processing," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 14, no. 4, pp. 336 –348, april 2006.

[3] L. Leem, H. Cho, J. Bau, Q. A. Jacobson, and S. Mitra, "Ersa: error resilient system architecture for probabilistic applications," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '10, 2010, pp. 1560–1565.

[4] C. Winstead, Y. Luo, E. Monzon, and A. Tejeda, "An error correction method for binary and multiple-valued logic," *Multiple-Valued Logic, IEEE International Symposium on*, pp. 105–110, 2011.

[5] C. Winstead and S. Howard, "Probabilistic LDPC-coded fault compensation technique for reliable nanoscale computing," *IEEE Transactions on Circuits and Systems II – Express Briefs*, vol. 56, no. 6, pp. 484–488, June 2009.

[6] Y. Tang, E. Boutillon, C. Jégo, and M. Jézéquel, "A new single-error correction scheme based on self-diagnosis residue number arithmetic," in *DASIP*, 2010, pp. 27–33.

[7] S. Yazdi, C.-H. Huang, and L. Dolecek, "Optimal design of a gallager b noisy decoder for irregular ldpc codes," *Communications Letters, IEEE*, vol. 16, no. 12, pp. 2052 –2055, december 2012.

[8] Y. Tang, C. Winstead, E. Boutillon, C. Jégo, and M. Jézéquel, "An ldpc decoding method for fault-tolerant digital logic," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, may 2012, pp. 3025 –3028.

[9] C. Winstead, Y. Tang, E. Boutillon, C. Jego, and M. Jezequel, "A space-time redundancy technique for embedded stochastic error correction," in *Turbo Codes and Iterative Information Processing (ISTC), 2012 7th International Symposium on*, aug. 2012, pp. 36 –40.

[10] R. Gallager, *Low Density Parity Check Codes*. MIT Press, 1963.

[11] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[12] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 599–618, 2001.

[13] D. Muller and W. Bertky, "A theory of asynchronous circuits," in *Proc. International Symposium on the Theory of Switching, Part 1*, 1959, pp. 204–243.

[14] W. Rao, A. Orailoglu, and R. Karri, "Logic mapping in crossbar-based nanoarchitectures," *Design Test of Computers, IEEE*, vol. 26, no. 1, pp. 68 –77, jan.-feb. 2009.